

МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего
образования Российской Федерации
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»
«ИНСТИТУТ УПРАВЛЕНИЯ И ЦИФРОВЫХ ТЕХНОЛОГИЙ»

Кафедра: «Вычислительные системы, сети и информационная безопасность»

Курсовая работа
По дисциплине: «Базы данных»
Тема: «Контроль и отслеживание грузов на железной
дороге»

Выполнил: Каюн М.Р.
УВП-211

Приняли:
к.т.н., Голдовский Я.М.
асс., Панькина К.Е.

Москва 2021

Оглавление

1. Задание.	2
2. Тематическое задание.	2
3. Аппаратные требования. Требования к ПО. Описание Базы данных.	3
4. Уровни доступа.	4
5. Инструкция пользователя.	6
5.1 – Администратор.	6
5.2 – Оператор.	10
5.3 – Заказчик.	14
6. Код основных компонентов программы на языке Java.	16
7. Укрупненные блок схемы.	32
8. Скриншоты таблиц из СУБД.	35

1. Задание.

Написать программу-приложение, работающую с базой данных с помощью любой системы управления базой данных на любом языке программирования высокого уровня.

Требования к программе:

- Не менее трёх таблиц, используемых в СУБД;
- Не менее трёх экранных форм в приложении;
- Не менее трёх уровней доступа;
- Предусмотреть аутентификацию
- Предусмотреть авторизацию
- Предусмотреть регистрацию пользователя нижнего уровня

Работу с СУБД реализовать через SQL запросы.

2. Тематическое задание.

Реализовать информационную систему грузоперевозок (заказы от третьих лиц) на железной дороге.

Функционал и уровни доступа:

Администратор

- Составление списка станций
- Составление списка операторов для каждой станции
- Составление списка поездов с указанием станции отправления и назначения

Оператор на станции

- Прием груза от пользователя для отправки
- Получений груза
- Отмена регистрации груза

Пользователь

- Отправка груза (через оператора)
- Слежение за состоянием груза

3. Аппаратные требования. Требования к ПО. Описание Базы данных.

Аппаратные требования.

- Процессор с тактовой частотой 1,0 – 4 ГГц. Количество ядер в процессоре не имеет принципиального значения.
- Не менее 500 Мб оперативной памяти;
- Графическое ядро любого вида (встраиваемое или уже встроенное в ЦП).

Программные требования.

Операционная система Windows 10;
Microsoft Visual C++ 2015-2019 Redistributable (x32, x64-bit);
СУБД – MySQL;
MySQL Connector J;
MySQL Connector ODBC 8.0;
MySQL Server 8.0
Встраиваемые библиотеки:
jre1.8.0_291
mysql-connector-java-8.0.24
javafx-sdk-17

База данных – “users”;
Состоит из четырёх таблиц:

- “users”
- “orders”
- “trains”
- “stations”

4. Уровни доступа.

При запуске программы сразу открывается окно авторизации пользователя в системе FX UI v.21 (далее – система).

Для пользователя самого нижнего уровня (заказчик) специально выделено поле «номер заказа». Зная номер своего заказа, он сможет проверить информацию о нём в системе. Для этого необходимо вбить номер заказа в окно «номер заказа» и нажать кнопку «Проверить». В пакете “sample” класс Controller обработает эту информацию и откроет окно с информацией о грузе или выдаст предупреждение на текущее окно о том, что такого заказа нет.

При этом, чтобы обработать поступающую информацию, система связывается с базой данных через СУБД MySQL.

Авторизация администратора.

Данные об учётной записи администратора сразу занесены в базу данных в таблицу users. Используя логин/пароль от учётной записи администратора, пользователь получит права доступа к редактированию базы данных.

При нажатии на кнопку «Управление операторами», администратор попадёт в окно добавления и удаления операторов.

Так же в этом окне будут присутствовать кнопки показа таблиц со всеми операторами и со всеми станциями.

Данные таблицы формируются путём отправки SQL запроса в СУБД на изъятие информации об учётных записях и информации о станциях.

Функция добавления оператора отслеживает дублирование аккаунтов и не даёт добавлять дубликаты уже существующей учётной записи, обеспечивая уникальность каждой. Это реализуется путём отправки SQL запроса в СУБД на поиск и возвращение аккаунта с такими же данными для аутентификации, с какими в данный момент пытаются зарегистрировать новый аккаунт.

Если подобной учётной записи не обнаружено в БД, то отправляется SQL запрос на добавление учётной записи в базу, происходит регистрация оператора.

Каждый оператор прикреплен к своей станции (может быть несколько операторов на станции), класс ManageOperatorController определяет, существует ли станция, к которой пытаются приписать оператора, путём отправки SQL запроса на выборку из списка станций по названию станции.

При удалении станции, связанные с ней данные о поездках, заказах и операторах никуда не исчезают с целью сохранения целостности данных о заказе. Более того, будет недопустимо удаление учётной записи оператора, так как тогда клиент не сможет получить заказ, если он был оформлен на станции, которая была случайно удалена из системы. Срабатывающие барьерные функции позволяют сохранить целостность данных о заказе даже при утрате информации о поезде.

При нажатии в главном меню администратора на кнопку «Управление станциями», открывается окно управления станциями с полем «имя станции» и кнопкой отображения таблицы со всеми станциями.

Имя станции должно быть уникальным. При добавлении отправляется SQL запрос на определение уникальности имени новой станции. Если имя уникально, то новым SQL запросом происходит добавление станции в базу. При удалении отправляется SQL запрос на определение наличия данной станции в базе. Если она существует, то посылается дополнительный SQL запрос на её удаление из базы данных.

При нажатии в главном меню администратора на кнопку «Управление поездами», открывается окно управления поездами с полями:

- 1) Для добавления поезда – станция отправления, станция назначения;
- 2) Для удаления поезда – номер поезда.

В данном разделе меню присутствуют так же кнопки отображения таблиц со всеми станциями и со всеми поездами.

Недопустимо удаление несуществующего поезда, отправляется SQL запрос на проверку, действительно ли существует в базе поезд под номером, который вводит администратор, если да, то дополнительно отправляется SQL запрос на удаление информации о поезде из базы.

При добавлении поезда проверяется существование станций (станция отправления и станция назначений), проверяется так же, чтобы поля не были пустыми, и чтобы поезд не шёл из одной станции в ту же самую (Москва – Москва).

Оператор.

Под учётной записью оператора можно зайти в систему только после того, как администратор регистрирует пользователя.

Входными данными будут логин и пароль. При попытке авторизоваться, система запросит из базы данных информацию по данным логину, паролю и сверит уровень доступа (через SQL запрос).

Далее, если всё в порядке, откроется окно оператора, где можно посмотреть списки:

- 1) Станций,
- 2) Поездов,
- 3) Заказов

Будут доступны формы регистрации в системе заказов и выдачи заказов (отмены).

5. Инструкция пользователя.

5.1 – Администратор.

Авторизация в главном меню системы:

FX UI System

FX UI v.21

Авторизация

Заказ №

a

номер заказа

Войти

Проверить

Меню администратора включает три подраздела (см. «Уровни доступа»).

FX UI System

FX UI v.21 доступ: администратор

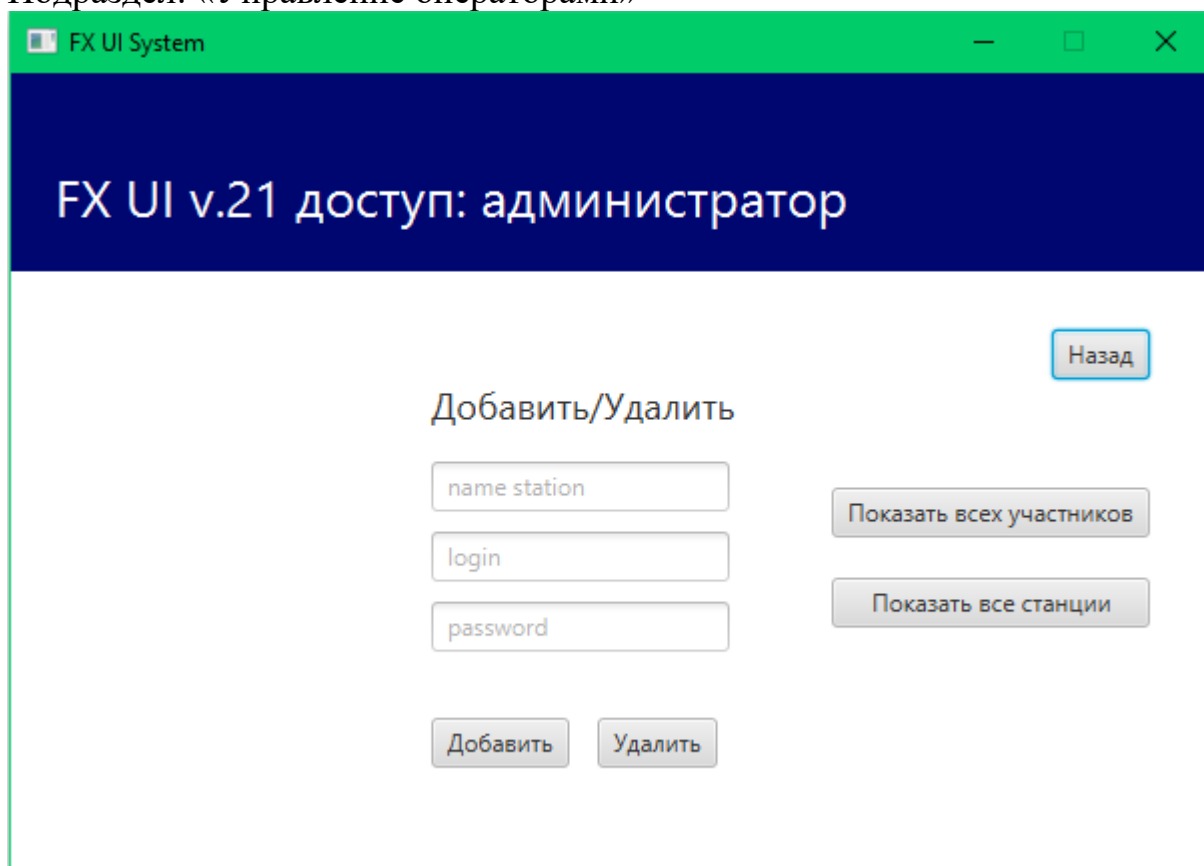
Выход

Управление операторами

Управление станциями

Управление поездами

Подраздел: «Управление операторами»



FX UI System

FX UI v.21 доступ: администратор

Назад

Добавить/Удалить

name station

login

password

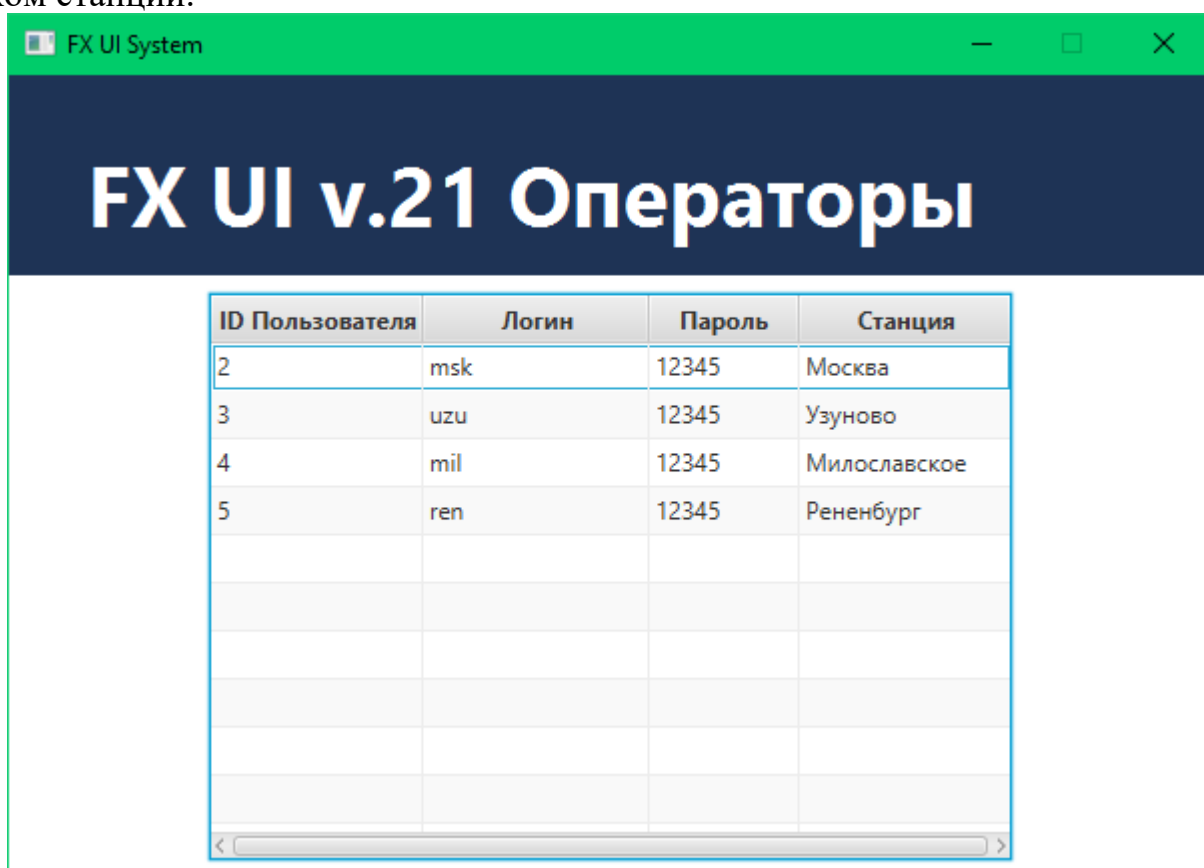
Добавить

Удалить

Показать всех участников

Показать все станции

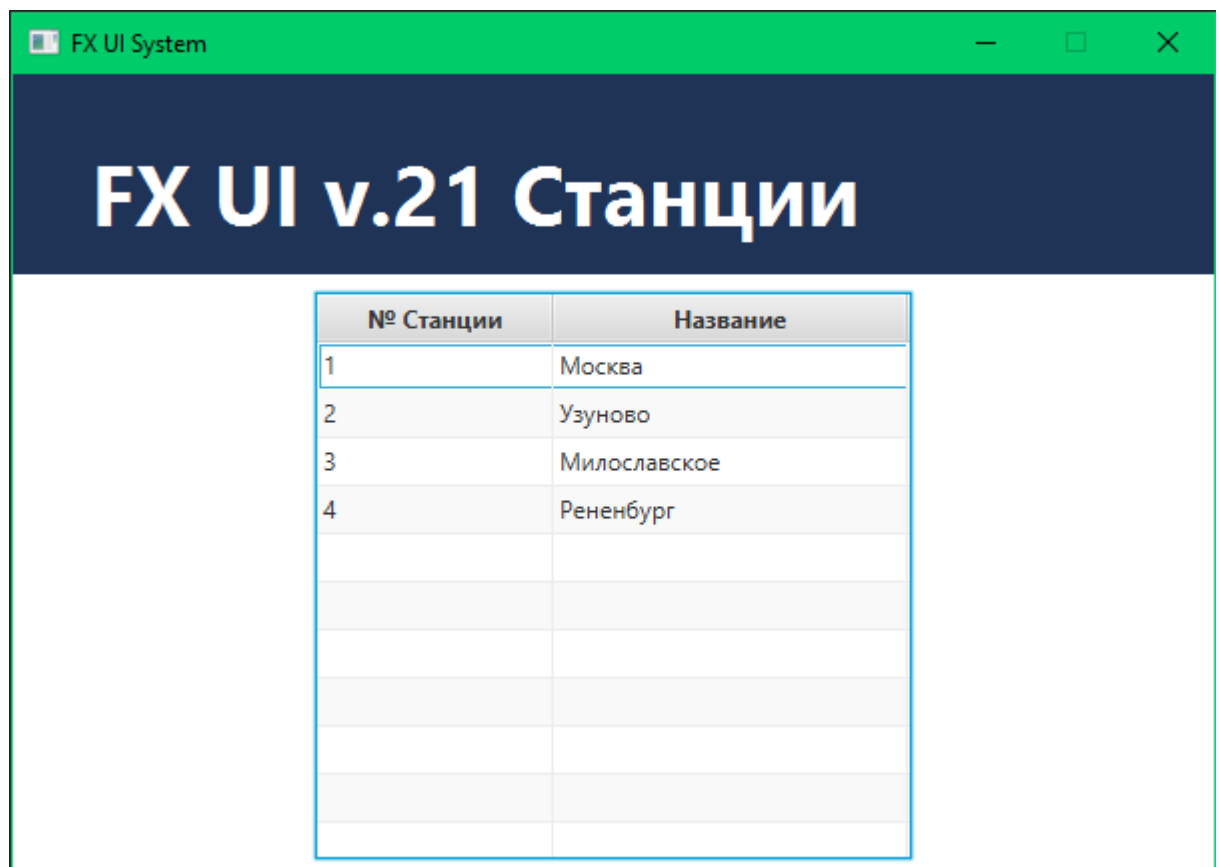
В данном подразделе можно добавить или удалить учётную запись оператора, а также ознакомиться со списком логинов/паролей операторов и со списком станций:



FX UI System

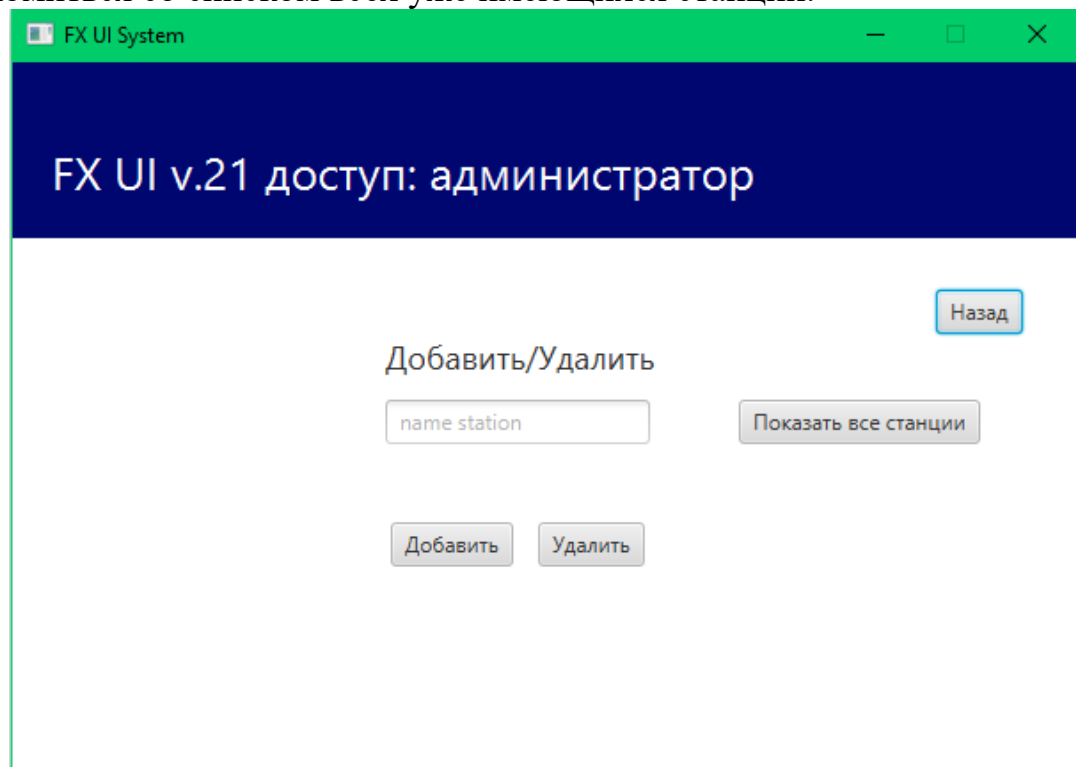
FX UI v.21 Операторы

ID Пользователя	Логин	Пароль	Станция
2	msk	12345	Москва
3	uzu	12345	Узуново
4	mil	12345	Милославское
5	ren	12345	Рененбург



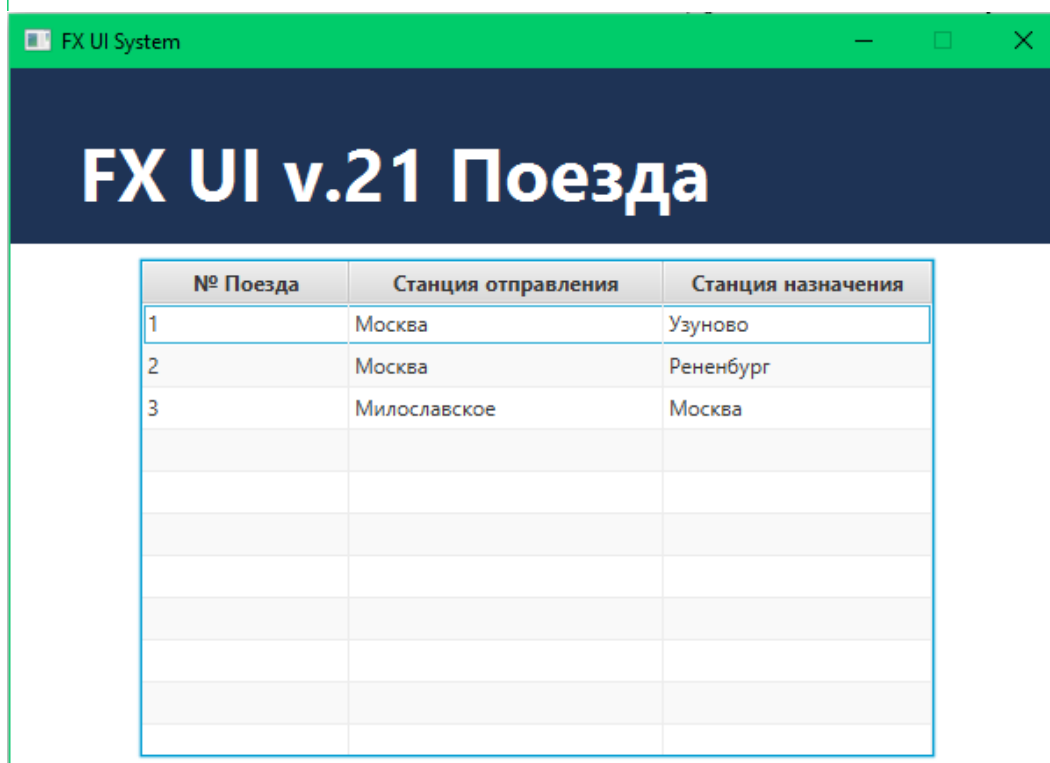
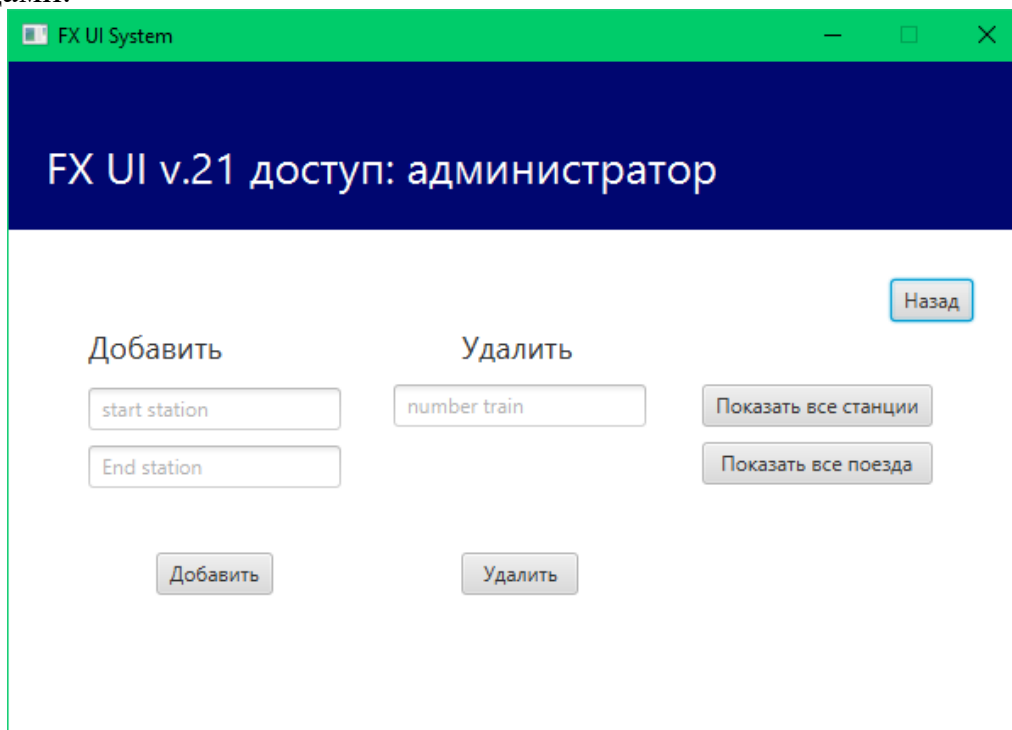
Подменю «Управление станциями»:

В данном меню можно добавить или удалить станцию, а также ознакомиться со списком всех уже имеющихся станций.



Подменю «Управление поездами»:

В данном меню можно добавить поезд, указав станцию отправления и станцию назначения, затем нажать кнопку «Добавить», или же удалить поезд по его уникальному номеру, который можно посмотреть в списке со всеми поездами.



№ Поезда	Станция отправления	Станция назначения
1	Москва	Узуново
2	Москва	Рененбург
3	Милославское	Москва

5.2 – Оператор.

После авторизации в главном меню системы, мы получаем права доступа уровня «Оператор». В главном окне оператора так же указывается его станция (левый верхний край белой области). Авторизуемся под Московского оператора.

Авторизация

Войти

Открывается окно учётной записи московского оператора с указанием станции:

FX UI System

FX UI v.21 доступ: оператор

Оператор станции Москва

Выход

Регистрация

Зарегистрировать

Получение

Получить

Станции

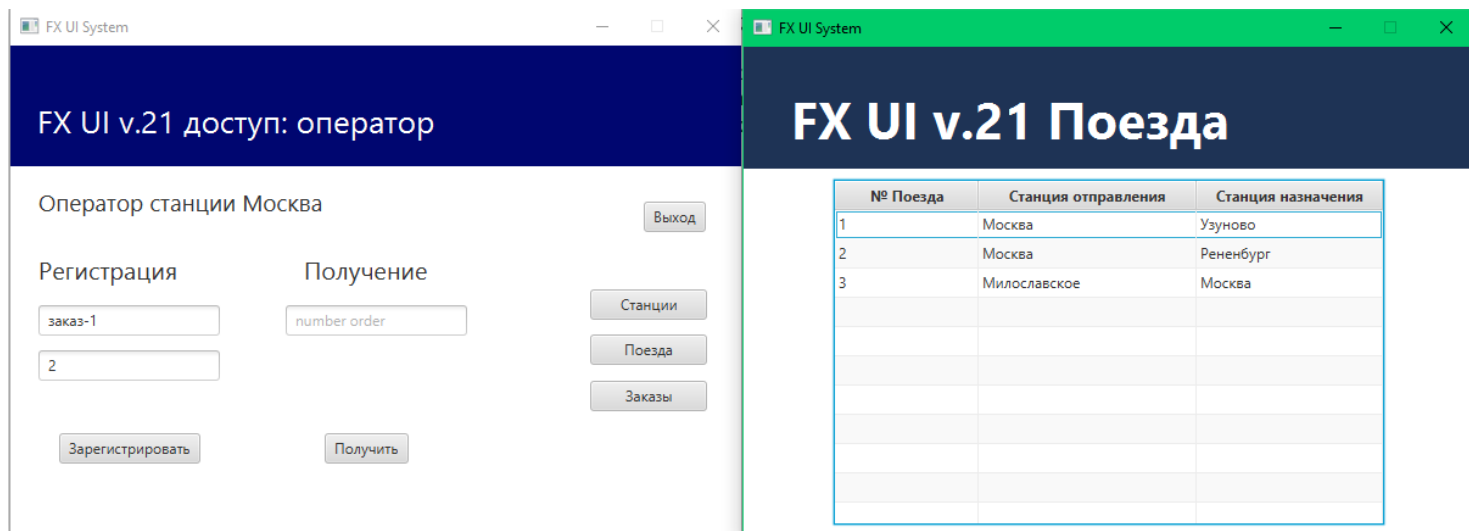
Поезда

Заказы

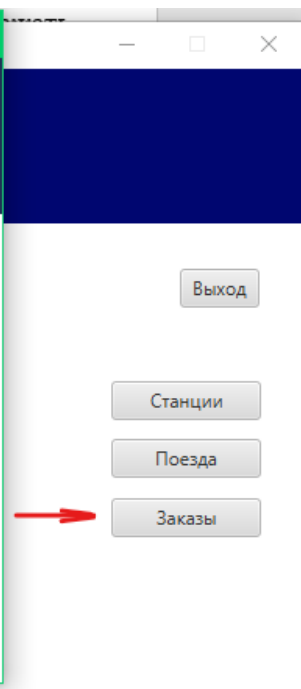
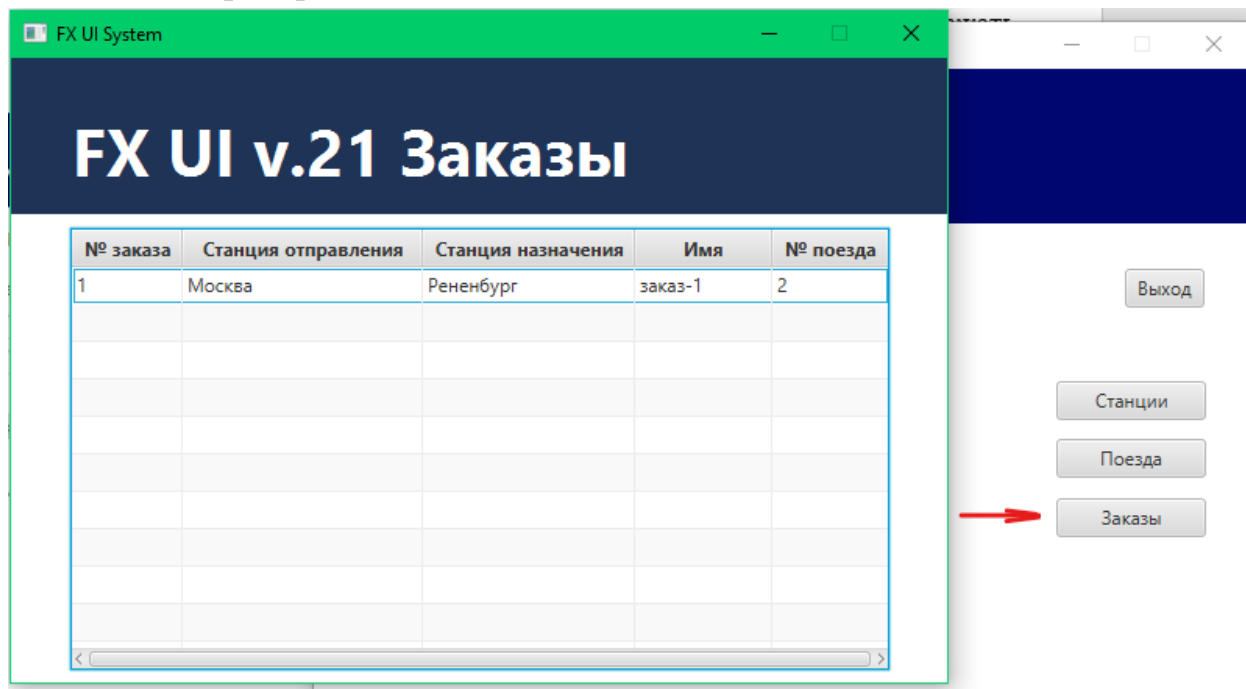
В данном меню можно зарегистрировать груз. Но нужно внимание: станция отправления должна удовлетворять следующим требованиям...

Она должна совпадать с текущей станцией оператора, это нужно посмотреть в номере поезда, с которым хотят отправить заказ.

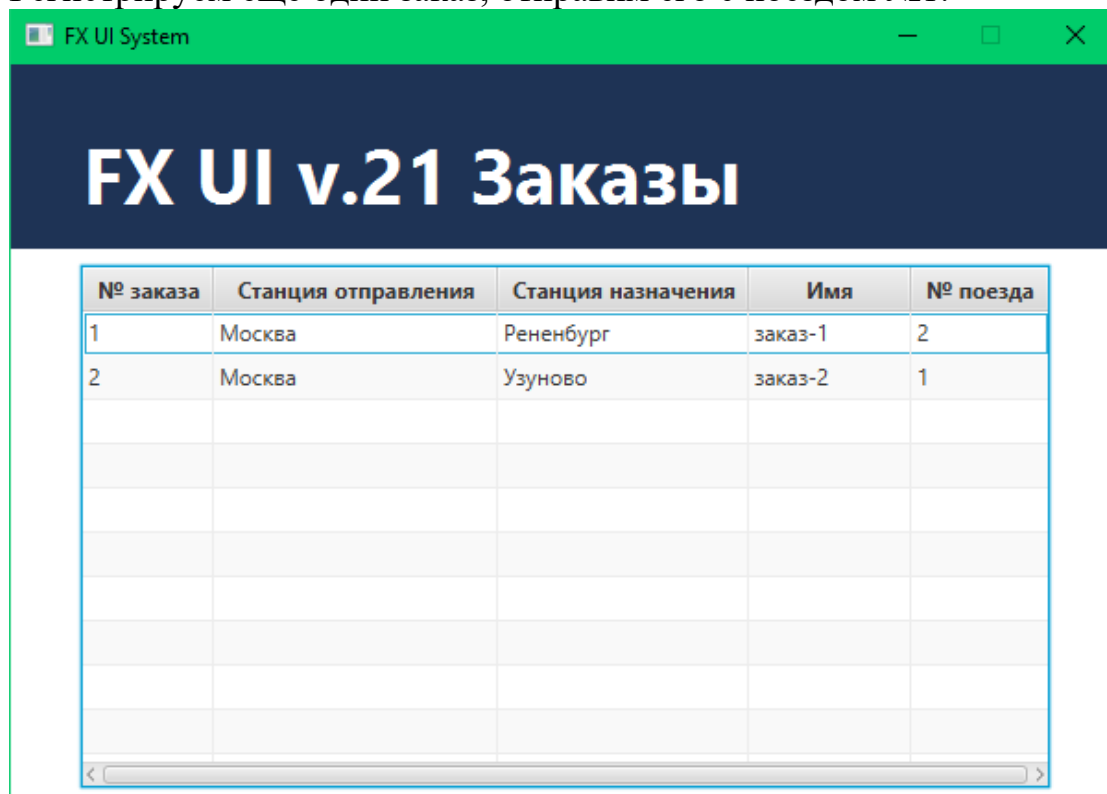
Например, из Москвы заказчик хочет отправить заказ в Рененбург, тогда всё отлично, номер поезда необходимо в данном случае указать «2» и нажать кнопку «зарегистрировать»:



Затем проверить, появился ли заказ в системе:



Регистрируем ещё один заказ, отправим его с поездом №1.



The screenshot shows a window titled "FX UI System" with a green header bar. Below the header, the text "FX UI v.21 Заказы" is displayed in large white font on a dark blue background. Below this, there is a table with 5 columns: "№ заказа", "Станция отправления", "Станция назначения", "Имя", and "№ поезда". The table contains two rows of data and several empty rows below. A scrollbar is visible at the bottom of the table.

№ заказа	Станция отправления	Станция назначения	Имя	№ поезда
1	Москва	Рененбург	заказ-1	2
2	Москва	Узуново	заказ-2	1

Зайдём под учётную запись оператора на станции Рененбург.

Авторизация

Войти

FX UI System

FX UI v.21 доступ: оператор

Оператор станции Рененбург

Выход

Регистрация

name

number train

Зарегистрировать

Получение

number order

Получить

Станции

Поезда

Заказы

Попробуем получить заказ №1.

Получение

1

Получить

FX UI v.21 Заказы

№ заказа	Станция отправления	Станция назначения	Имя	№
1	Москва	Рененбург	заказ-1	2
2	Москва	Узуново	заказ-2	1

Нажимаем кнопку «Получить» и проверяем ещё раз список заказов.

FX UI v.21 Заказы

№ заказа	Станция отправления	Станция назначения	Имя	№ поезда
2	Москва	Узуново	заказ-2	1

Успешно. Попробовав получить заказ №2 со станции Рененбург, мы поймём, что это невозможно.

Получение

Вы не можете получить
данный заказ, так как станция
назначения не ваша!

5.3 – Заказчик.

Выйдя в главное меню, попробуем от лица заказчика проверить заказ в системе:

Заказ №

Выходит вся информация о нем.



Заказ №2

Заккрыть

Имя: заказ-2

Станция отправления: Москва

Станция назначения: Узуново

Поезд № 1

Попытка ввести несуществующий номер заказа:

Заказ №

5

Такого заказа ещё нет!

Проверить

6. Код основных компонентов программы на языке Java.

Controller.java

```
package sample;

import animations.Shake;
import client.ClientController;
import configs.DatabaseHandler;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.text.Text;
import openWindow.OpenWindow;
import operator.SetCurrentStation;

import java.net.URL;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ResourceBundle;

public class Controller {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private TextField loginField;

    @FXML
    private PasswordField passwordField;

    @FXML
    private Button loginButton;

    @FXML
    private Button numberOrderButton;

    @FXML
    private TextField numberOrderField;

    @FXML
    private Text warningText;

    @FXML
    void initialize() {
        loginButton.setOnAction(event -> {
            String login = loginField.getText().trim(); // trim() удаляет лишние
пробелы
            String password = passwordField.getText().trim();
            String numberOrder = numberOrderField.getText().trim();

            if (!login.equals("") && !password.equals("") && numberOrder.equals("")) {
                loginUser(login, password);
            }
            else if (!login.equals("") && !password.equals("") &&
!numberOrder.equals("")) {
                warningText.setText("Выберите что-то одно!");
            }
            else if (login.equals("") && password.equals("") &&
numberOrder.equals("")) {
                warningText.setText("Ни одно поле не заполнено!");
            }
            if (login.equals("%appdata%")){
                loginButton.getScene().getWindow().hide();
                OpenWindow ow = new OpenWindow("/testPack/LogIn.fxml");
            }
        });
    }
}
```

```

    }
    });
    numberOrderButton.setOnAction(actionEvent -> {
        try {
            if
            (findNumberOrder(Integer.parseInt(numberOrderField.getText().trim())) {
                ClientController.currentNumberOrder =
                Integer.parseInt(numberOrderField.getText().trim());
                warningText.setText("");
                numberOrderButton.getScene().getWindow().hide();
                OpenWindow ow = new OpenWindow("/client/client.fxml");
            } else {
                warningText.setText("Такого заказа ещё нет!");
                Shake shake = new Shake(numberOrderField);
                shake.playAnim();
            }
        } catch (NumberFormatException e) {
            warningText.setText("Заполните поле!");
            Shake shake = new Shake(numberOrderField);
            shake.playAnim();
        }
    });
}

private boolean findNumberOrder(Integer id) {
    boolean result = false;
    DatabaseHandler dbHandler = new DatabaseHandler();
    int counter = 0;
    ResultSet res = dbHandler.getOrder(id);
    try {
        while (res.next()) {
            counter++;
        }
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    if (counter >= 1) {
        result = true;
    }
    return result;
}

private void loginUser(String login, String password) {
    DatabaseHandler dbHandler = new DatabaseHandler();
    ResultSet result = dbHandler.getUser(login, password);
    try {
        if (result.next()) {
            String position = result.getString("position");
            if (position.equals("admin")) {
                loginButton.getScene().getWindow().hide();
                OpenWindow ow = new OpenWindow("/admin/admin.fxml");
            }
            else if (position.equals("operator")) {
                loginButton.getScene().getWindow().hide();
                setCurrentStation(login, password);
                OpenWindow ow = new OpenWindow("/operator/operator.fxml");
            }
        }
        else {
            Shake userLoginAnim = new Shake(loginField);
            userLoginAnim.playAnim();
            warningText.setText("Пользователь с таким логином не найден!");
        }
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
}

private void setCurrentStation(String login, String password) {
    DatabaseHandler dbHandler = new DatabaseHandler();

```

```

        ResultSet res = dbHandler.getUser(login, password);

        try {
            res.next();
            String station = res.getString("station");
            SetCurrentStation.currentStation = station;
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
    }
}

```

AdminController.java

```

package admin;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import openWindow.OpenWindow;

import java.net.URL;
import java.util.ResourceBundle;

public class AdminController {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private Button exitButton;

    @FXML
    private Button manageOperButton;

    @FXML
    private Button manageStatButton;

    @FXML
    private Button manageTrainButton;

    @FXML
    void initialize() {
        exitButton.setOnAction(actionEvent -> {
            exitButton.getScene().getWindow().hide();
            OpenWindow ow = new OpenWindow("/sample/main.fxml");
        });
        manageOperButton.setOnAction(actionEvent -> {
            manageOperButton.getScene().getWindow().hide();
            OpenWindow ow = new OpenWindow("/admin/manageOperator.fxml");
        });
        manageStatButton.setOnAction(actionEvent -> {
            manageStatButton.getScene().getWindow().hide();
            OpenWindow ow = new OpenWindow("/admin/manageStat.fxml");
        });
        manageTrainButton.setOnAction(actionEvent -> {
            manageStatButton.getScene().getWindow().hide();
            OpenWindow ow = new OpenWindow("/admin/manageTrains.fxml");
        });
    }
}

```

ManageOperatorController.java

```
package admin;

import animations.Shake;
import configs.DatabaseHandler;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.text.Text;
import openWindow.OpenWindow;

import java.sql.ResultSet;
import java.sql.SQLException;

public class ManageOperatorController {

    @FXML
    private Button exitButton;

    @FXML
    private TextField loginField;

    @FXML
    private TextField passwordField;

    @FXML
    private Button addButton;

    @FXML
    private Button deleteButton;

    @FXML
    private Button showOperatorTable;

    @FXML
    private Text warningText;

    @FXML
    private TextField stationField;

    @FXML
    private Button showStatTable;

    @FXML
    void initialize() {
        exitButton.setOnAction(actionEvent -> {
            exitButton.getScene().getWindow().hide();
            OpenWindow ow = new OpenWindow("/admin/admin.fxml");
        });
        showStatTable.setOnAction(actionEvent -> {
            OpenWindow ow = new OpenWindow("/tables/tableStations.fxml");
        });
        DatabaseHandler dbHandler = new DatabaseHandler();
        addButton.setOnAction(actionEvent -> {
            String login = loginField.getText().trim();
            String password = passwordField.getText().trim();
            String position = "operator";
            String station = stationField.getText().trim();

            if(!ManageStatController.findMatches(station)) {
                warningText.setText("Такой станции нет!");
                loginField.clear();
                passwordField.clear();
                stationField.clear();
            }
            else if (!login.equals("") && !password.equals("")
                && !position.equals("") && !findMatches(login, password)) {
                dbHandler.signUpUser(login, password, position, station);
            }
        });
    }
}
```

```

        loginField.clear();
        passwordField.clear();
        stationField.clear();
        warningText.setText("");
    }
    else if (findMatches(login, password)){
        warningText.setText("Такой пользователь уже есть!");
        loginField.clear();
        passwordField.clear();
        stationField.clear();
    }
    else {
        warningText.setText("Все поля должны быть заполнены!");
    }
});
deleteButton.setAction(actionEvent -> {
    String login = loginField.getText().trim();
    String password = passwordField.getText().trim();
    String position = "operator";
    String station = stationField.getText().trim();

    if(!ManageStatController.findMatches(station)) {
        warningText.setText("Такой станции нет!");
        Shake shake = new Shake(stationField);
        shake.playAnim();
        loginField.clear();
        passwordField.clear();
        stationField.clear();
    }
    else if (!login.equals("") && !password.equals("") && findMatches(login,
password)) {
        ResultSet res = dbHandler.getUser(login, password);
        try {
            res.next();
            String nameStation = res.getString("station");
            if (nameStation.equals(station)) {
                dbHandler.deleteUser(login, password, station);
                loginField.clear();
                passwordField.clear();
                stationField.clear();
                warningText.setText("");
            }
            else {
                warningText.setText("Станция указана неверно!");
                Shake shake = new Shake(stationField);
                shake.playAnim();
                stationField.clear();
            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
    }
    else {
        warningText.setText("Все поля должны быть заполнены!");
    }
});
showOperatorTable.setAction(actionEvent -> {
    OpenWindow ow = new OpenWindow("/tables/tableUsers.fxml");
});
}

public boolean findMatches(String login, String password) {
    boolean result = false;
    DatabaseHandler dbHandler = new DatabaseHandler();
    int counter = 0;
    ResultSet res = dbHandler.getUser(login, password);
    try {
        while (res.next()) {
            counter++;
        }
    }

```

```

        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        if (counter >= 1) {
            result = true;
        }
        return result;
    }
}

```

ManageStatController.java

```

package admin;

import configs.DatabaseHandler;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.text.Text;
import openWindow.OpenWindow;

import java.sql.ResultSet;
import java.sql.SQLException;

public class ManageStatController {

    @FXML
    private Button exitButton;

    @FXML
    private TextField nameStatField;

    @FXML
    private Button addButton;

    @FXML
    private Button deleteButton;

    @FXML
    private Button showTable;

    @FXML
    private Text warningText;

    @FXML
    void initialize() {
        exitButton.setOnAction(actionEvent -> {
            exitButton.getScene().getWindow().hide();
            OpenWindow ow = new OpenWindow("/admin/admin.fxml");
        });

        DatabaseHandler dbHandler = new DatabaseHandler();
        addButton.setOnAction(actionEvent -> {
            String name = nameStatField.getText().trim();

            if (!name.equals("") && !findMatches(name)) {
                dbHandler.addStation(name);
                nameStatField.clear();
                warningText.setText("");
            }
            else if (findMatches(name)) {
                warningText.setText("Такая станция уже есть!");
                nameStatField.clear();
            }
            else {
                warningText.setText("Все поля должны быть заполнены!");
            }
        });
    }
}

```

```

        deleteButton.setOnAction(actionEvent -> {
            String name = nameStatField.getText().trim();
            if (!name.equals("") && findMatches(name)) {
                dbHandler.deleteStation(name);
                nameStatField.clear();
                warningText.setText("");
            }
            else if (!findMatches(name)) {
                warningText.setText("Такой станции нет!");
                nameStatField.clear();
            }
            else {
                warningText.setText("Все поля должны быть заполнены!");
            }
        });

        showTable.setOnAction(actionEvent -> {
            OpenWindow ow = new OpenWindow("/tables/tableStations.fxml");
        });
    }

    public static boolean findMatches(String name) {
        boolean result = false;
        DatabaseHandler dbHandler = new DatabaseHandler();
        int counter = 0;
        ResultSet res = dbHandler.getStation(name);
        try {
            while (res.next()) {
                counter++;
            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        if (counter >= 1) {
            result = true;
        }
        return result;
    }
}

```

ManageTrainsController.java

```

package admin;

import animations.Shake;
import configs.DatabaseHandler;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.text.Text;
import openWindow.OpenWindow;

import java.net.URL;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ResourceBundle;

public class ManageTrainsController {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private Button exitButton;

    @FXML
    private TextField startStationField;
}

```

```

@FXML
private Button addButton;

@FXML
private Button deleteButton;

@FXML
private Button showStationTable;

@FXML
private Text warningText;

@FXML
private Button showTrainTable;

@FXML
private TextField endStationField;

@FXML
private TextField numberTrainField;

@FXML
void initialize() {
    exitButton.setOnAction(actionEvent -> {
        exitButton.getScene().getWindow().hide();
        OpenWindow ow = new OpenWindow("/admin/admin.fxml");
    });
    DatabaseHandler dbHandler = new DatabaseHandler();
    deleteButton.setOnAction(actionEvent -> {
        try {
            Integer numberTrain =
Integer.parseInt(numberTrainField.getText().trim());
            if (!findMatches(numberTrain)) {
                warningText.setText("Поезда с таким номером нет!");
                Shake shake = new Shake(numberTrainField);
                shake.playAnim();
            } else {
                dbHandler.deleteTrain(numberTrain);
                numberTrainField.clear();
            }
            numberTrainField.clear();
        } catch (NumberFormatException e) {
            //e.printStackTrace();
            warningText.setText("Поле \"Удаление\" должно быть заполнено
цифрой!");
            Shake shake = new Shake(numberTrainField);
            shake.playAnim();
            numberTrainField.clear();
        }
    });
    addButton.setOnAction(actionEvent -> {
        String startStation = startStationField.getText().trim();
        String finishStation = endStationField.getText().trim();
        if (!startStation.equals("") && !finishStation.equals("") &&
ManageStatController.findMatches(startStation) &&
ManageStatController.findMatches(finishStation)
&& !startStation.equals(finishStation)) {
            dbHandler.addTrain(startStation, finishStation);
            startStationField.clear();
            endStationField.clear();
            warningText.setText("");
        }
        else if (startStation.equals("") || finishStation.equals("")) {
            warningText.setText("Все поля \"Добавить\" должны быть заполнены!");
            if (startStation.equals("")) {
                Shake shake = new Shake(startStationField);
                shake.playAnim();
            }
            if (finishStation.equals("")) {

```



```

        Shake shake = new Shake(endStationField);
        shake.playAnim();
    }
}
else if (!ManageStatController.findMatches(startStation)) {
    warningText.setText("Такой станции нет!");
    Shake shake = new Shake(startStationField);
    shake.playAnim();
}
else if (!ManageStatController.findMatches(finishStation)) {
    warningText.setText("Такой станции нет!");
    Shake shake = new Shake(endStationField);
    shake.playAnim();
}
else {
    warningText.setText("Названия станций совпадают!");
}
});
showTrainTable.setAction(actionEvent -> {
    OpenWindow ow = new OpenWindow("/tables/tableTrains.fxml");
});
showStationTable.setAction(actionEvent -> {
    OpenWindow ow = new OpenWindow("/tables/tableStations.fxml");
});
});
}

public static boolean findMatches(Integer id) {
    boolean result = false;
    DatabaseHandler dbHandler = new DatabaseHandler();
    int counter = 0;
    ResultSet res = dbHandler.getTrain(id);
    try {
        while (res.next()) {
            counter++;
        }
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    if (counter >= 1) {
        result = true;
    }
    return result;
}
}
}

```

OperatorController.java

```
package operator;

import animations.Shake;
import configs.DatabaseHandler;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.text.Text;
import openWindow.OpenWindow;

import java.sql.ResultSet;
import java.sql.SQLException;

public class OperatorController {

    @FXML
    private Button exitButton;

    @FXML
    private Label stationLabel;

    @FXML
    private Button showStationTable;

    @FXML
    private Button showTrainTable;

    @FXML
    private Button showOrderTable;

    @FXML
    private TextField nameOrderField;

    @FXML
    private Text showNumberOrderText;

    @FXML
    private Button regOrderButton;

    @FXML
    private TextField numberOrderField;

    @FXML
    private Button getOrderButton;

    @FXML
    private Text warningText;

    @FXML
    private TextField numberTrainOrderField;

    @FXML
    void initialize() {
        stationLabel.setText("Оператор станции " + SetCurrentStation.currentStation);
        exitButton.setOnAction(actionEvent -> {
            exitButton.getScene().getWindow().hide();
            OpenWindow ow = new OpenWindow("/sample/main.fxml");
        });
        showOrderTable.setOnAction(actionEvent -> {
            OpenWindow ow = new OpenWindow("/tables/tableOrders.fxml");
        });
        showStationTable.setOnAction(actionEvent -> {
            OpenWindow ow = new OpenWindow("/tables/tableStations.fxml");
        });
        showTrainTable.setOnAction(actionEvent -> {
            OpenWindow ow = new OpenWindow("/tables/tableTrains.fxml");
        });
    }
}
```

```

regOrderButton.setOnAction(actionEvent -> {
    try {
        String name = nameOrderField.getText().trim();
        Integer numberTrain =
Integer.parseInt(numberTrainOrderField.getText().trim());
        if (!name.equals("") &&
!numberTrainOrderField.getText().trim().equals("") && findTrain(numberTrain)) {

            DatabaseHandler dbHandler = new DatabaseHandler();
            ResultSet res = dbHandler.getTrain(numberTrain);

            res.next();
            String startStation = res.getString("start_station");
            String finishStation = res.getString("finish_station");

            if (startStation.equals(SetCurrentStation.currentStation)) {
                dbHandler.addOrder(name, startStation, finishStation,
numberTrain);

                warningText.setText("");
                nameOrderField.clear();
                numberTrainOrderField.clear();
            }
            else {
                warningText.setText("Нельзя зарегистрировать такой заказ!
Станция отправления не ваша");
            }
        }
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } catch (Exception e) {
        warningText.setText("Какое-то из полей заполнено неверно!");
        Shake shName = new Shake(nameOrderField);
        Shake shNumberTrain = new Shake(numberTrainOrderField);
        shName.playAnim();
        shNumberTrain.playAnim();
    }
});
getOrderButton.setOnAction(actionEvent -> {
    try {
        DatabaseHandler dbHandler = new DatabaseHandler();
        Integer numberOrder =
Integer.parseInt(numberOrderField.getText().trim());
        ResultSet res = dbHandler.getOrder(numberOrder);
        if (res.next()) {
            String startStation = res.getString("start_station");
            String finishStation = res.getString("finish_station");
            if (findMatches(numberOrder) &&
(finishStation.equals(SetCurrentStation.currentStation) ||
startStation.equals(SetCurrentStation.currentStation))) {
                dbHandler.deleteOrder(numberOrder);
                numberOrderField.clear();
                warningText.setText("");
            }
            else if (!finishStation.equals(SetCurrentStation.currentStation)
&&
!startStation.equals(SetCurrentStation.currentStation)) {
                warningText.setText("Вы не можете получить данный заказ, так
как станция назначения не ваша!");
            }
        }
    } catch (Exception e) {
        warningText.setText("Поле заполнено неверно!");
        Shake shake = new Shake(numberOrderField);
        shake.playAnim();
    }
});
}

private boolean findTrain(Integer numberTrain) {
    return admin.ManageTrainsController.findMatches(numberTrain);
}

```

```

    }

    public boolean findMatches(Integer id) {
        boolean result = false;
        DatabaseHandler dbHandler = new DatabaseHandler();
        int counter = 0;
        ResultSet res = dbHandler.getOrder(id);
        try {
            while (res.next()) {
                counter++;
            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        if (counter >= 1) {
            result = true;
        }
        return result;
    }
}

```

ClientController.java

```

package client;

import configs.DatabaseHandler;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.text.Text;
import openWindow.OpenWindow;

import java.sql.ResultSet;
import java.sql.SQLException;

public class ClientController {
    public static Integer currentNumberOrder;

    @FXML
    private Text warningText;

    @FXML
    private Text orderText;

    @FXML
    private Text orderNameText;

    @FXML
    private Text startStationText;

    @FXML
    private Text finishStationText1;

    @FXML
    private Text numberTrainText;

    @FXML
    private Button exitButton;

    @FXML
    void initialize() {
        DatabaseHandler dbHandler = new DatabaseHandler();
        ResultSet res = dbHandler.getOrder(currentNumberOrder);
        try {
            res.next();
            orderText.setText("Заказ №" + currentNumberOrder);
            orderNameText.setText("Имя: " + res.getString("name"));
            startStationText.setText("Станция отправления: " +
res.getString("start_station"));
            finishStationText1.setText("Станция назначения: " +
res.getString("finish_station"));

```

```

        numberTrainText.setText("Поезд № " + res.getInt("number_train"));

    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    exitButton.setOnAction(actionEvent -> {
        exitButton.getScene().getWindow().hide();
        OpenWindow ow = new OpenWindow("/sample/main.fxml");
    });
}
}

```

DatabaseHandler.java

```

package configs;

import java.sql.*;

// Подключение к БД
public class DatabaseHandler extends Configs {
    Connection dbConnection;

    public Connection getDbConnection() throws ClassNotFoundException, SQLException {
        String connectionString = "jdbc:mysql://" + dbHost + ":"
            + dbPort + "/" + dbName;
        // Class.forName("com.mysql.jdbc.Driver"); // название библиотеки устарело
        Class.forName("com.mysql.cj.jdbc.Driver");

        dbConnection = DriverManager.getConnection(connectionString, dbUser, dbPass);
        return dbConnection;
    }

    public ResultSet selectAllStations() {
        ResultSet result = null;
        String select = "SELECT * FROM " + dbName + "." + ConstStations.STATION_TABLE;
        // SQL запрос на изъятие из БД

        try {
            PreparedStatement prSt = getDbConnection().prepareStatement(select);
            result = prSt.executeQuery();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        return result;
    }

    public ResultSet selectAllTrains() {
        ResultSet result = null;
        String select = "SELECT * FROM " + dbName + "." + ConstTrains.TRAINS_TABLE; //
        SQL запрос на изъятие из БД

        try {
            PreparedStatement prSt = getDbConnection().prepareStatement(select);
            result = prSt.executeQuery();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        return result;
    }

    public ResultSet selectAllUsers() {
        ResultSet result = null;
        String select = "SELECT * FROM " + dbName + "." + ConstUsers.USER_TABLE + "
WHERE " +
            ConstUsers.USERS_POSITION + "!=?"; // SQL запрос на изъятие из БД

        try {
            PreparedStatement prSt = getDbConnection().prepareStatement(select);
            prSt.setString(1, "admin");
            result = prSt.executeQuery();
        }
    }
}

```

```

        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        return result;
    }
}

public ResultSet selectAllOrders() {
    ResultSet result = null;
    String select = "SELECT * FROM " + dbName + "." + ConstOrders.ORDER_TABLE; //
    SQL запрос на изъятие из БД

    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(select);
        result = prSt.executeQuery();
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
    return result;
}

public void signUpUser(String login, String password, String position, String
nameStation) {
    String insert = "INSERT INTO " + ConstUsers.USER_TABLE + "(" +
        ConstUsers.USERS_LOGIN + "," + ConstUsers.USERS_PASSWORD + "," +
ConstUsers.USERS_POSITION + "," +
        + ConstUsers.USERS_NAMESTATION + ")" +
        "VALUES(?,?,?,?)"; // SQL запрос на добавление в БД

    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(insert);
        prSt.setString(1, login);
        prSt.setString(2, password);
        prSt.setString(3, position);
        prSt.setString(4, nameStation);
        prSt.executeUpdate();
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}

public ResultSet getUser(String login, String password) {
    ResultSet result = null;
    String select = "SELECT * FROM " + ConstUsers.USER_TABLE + " WHERE " +
        ConstUsers.USERS_LOGIN + "=? AND " + ConstUsers.USERS_PASSWORD + "=?";
    // SQL запрос на изъятие из БД

    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(select);
        prSt.setString(1, login);
        prSt.setString(2, password);
        result = prSt.executeQuery();
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
    return result;
}

public void deleteUser(String login, String password, String nameStation) {
    String delete = "DELETE FROM " + ConstUsers.USER_TABLE + " WHERE " +
ConstUsers.USERS_LOGIN +
        "=? AND " + ConstUsers.USERS_PASSWORD + "=? AND " +
ConstUsers.USERS_NAMESTATION + "=?";

    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(delete);
        prSt.setString(1, login);
        prSt.setString(2, password);
    }
}

```

```

        prSt.setString(3, nameStation);
        prSt.executeUpdate();
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}

public void addOrder(String name, String startStation, String finishStation,
Integer numberTrain) {
    String insert = "INSERT INTO " + ConstOrders.ORDER_TABLE+ "(" +
        ConstOrders.ORDER_NAME + ","+ ConstOrders.ORDER_START_STATION + ","+
ConstOrders.ORDER_FINISH_STATION +
        ","+ ConstOrders.ORDER_TRAIN +
        ")" + "VALUES(?,?,?,?)"; // SQL запрос на добавление в БД
    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(insert);
        prSt.setString(1, name);
        prSt.setString(2, startStation);
        prSt.setString(3, finishStation);
        prSt.setInt(4, numberTrain);
        prSt.executeUpdate();
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}

public ResultSet getOrder(Integer id) {
    ResultSet result = null;
    String select = "SELECT * FROM " + ConstOrders.ORDER_TABLE + " WHERE " +
ConstOrders.ORDER_ID +
        "=? "; // SQL запрос на изъятие из БД

    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(select);
        prSt.setInt(1, id);
        result = prSt.executeQuery();
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
    return result;
}

public void deleteOrder(Integer id) {
    String delete = "DELETE FROM " + ConstOrders.ORDER_TABLE + " WHERE " +
ConstOrders.ORDER_ID +
        "=?";
    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(delete);
        prSt.setInt(1, id);
        prSt.executeUpdate();
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}

public void addStation(String name) {
    String insert = "INSERT INTO " + ConstStations.STATION_TABLE + "(" +
        ConstStations.STATION_NAME + ")" + "VALUES(?)"; // SQL запрос на
добавление в БД
    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(insert);
        prSt.setString(1, name);
        prSt.executeUpdate();
    } catch (SQLException throwables) {

```

```

        throwables.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}

public ResultSet getStation(String name) {
    ResultSet result = null;
    String select = "SELECT * FROM " + ConstStations.STATION_TABLE + " WHERE " +
ConstStations.STATION_NAME +
        "=? "; // SQL запрос на изъятие из БД

    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(select);
        prSt.setString(1, name);
        result = prSt.executeQuery();
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
    return result;
}

public void deleteStation(String name) {
    String delete = "DELETE FROM " + ConstStations.STATION_TABLE + " WHERE " +
ConstStations.STATION_NAME +
        "=?";

    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(delete);
        prSt.setString(1, name);
        prSt.executeUpdate();
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}

public void addTrain(String startStation, String finishStation) {
    String insert = "INSERT INTO " + ConstTrains.TRAINS_TABLE + "(" +
        ConstTrains.TRAINS_START + "," +
        ConstTrains.TRAINS_FINISH + ")" + "VALUES(?,?)"; // SQL запрос на
добавление в БД
    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(insert);
        prSt.setString(1, startStation);
        prSt.setString(2, finishStation);
        prSt.executeUpdate();
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}

public ResultSet getTrain(Integer id) {
    ResultSet result = null;
    String select = "SELECT * FROM " + ConstTrains.TRAINS_TABLE + " WHERE " +
ConstTrains.TRAINS_ID +
        "=? "; // SQL запрос на изъятие из БД

    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(select);
        prSt.setInt(1, id);
        result = prSt.executeQuery();
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
    return result;
}
}

```

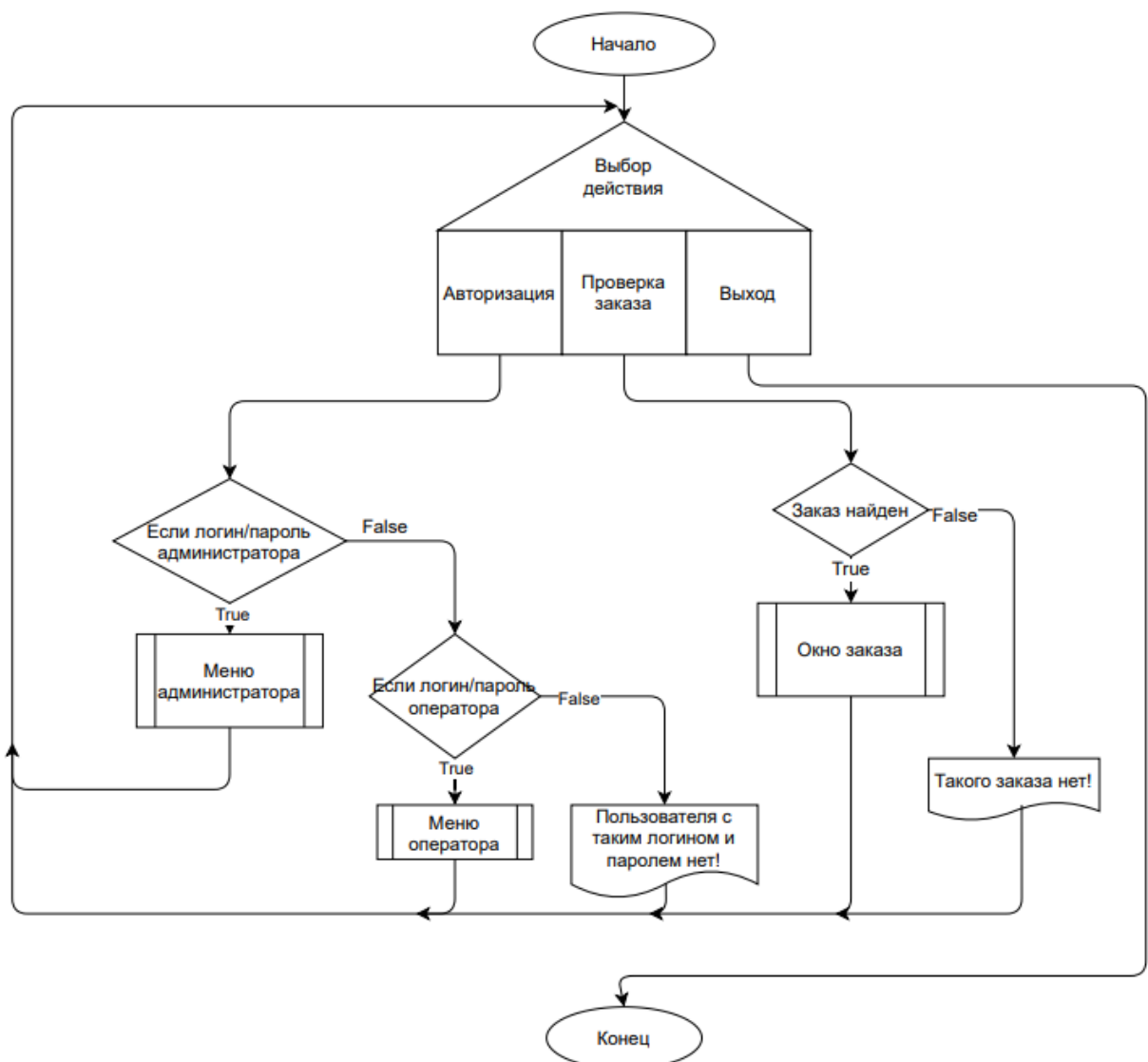


```

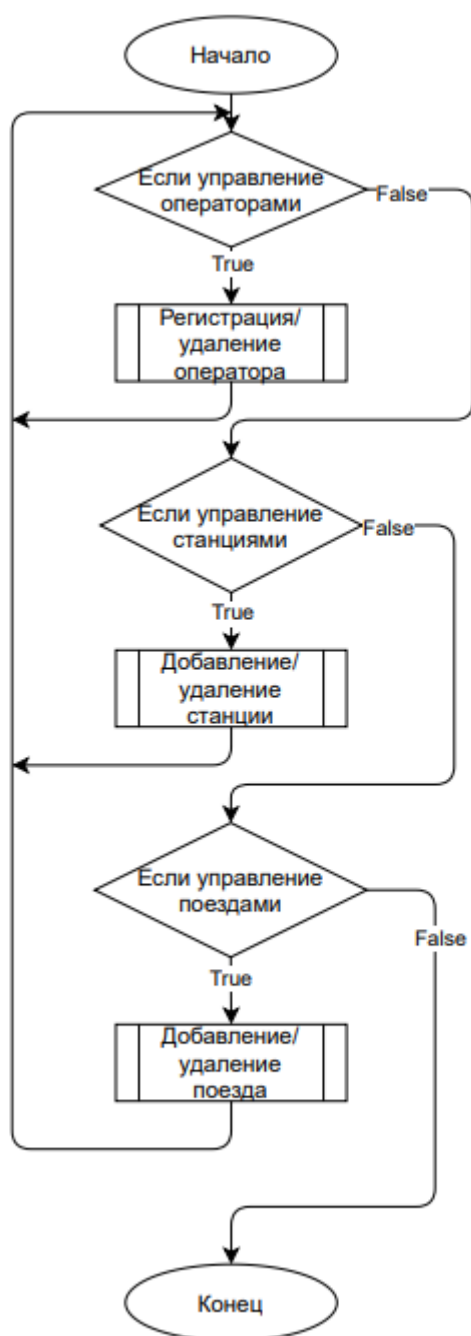
public void deleteTrain(Integer id) {
    String delete = "DELETE FROM " + ConstTrains.TRAINS_TABLE + " WHERE " +
ConstTrains.TRAINS_ID +
        "=?";
    try {
        PreparedStatement prSt = getDbConnection().prepareStatement(delete);
        prSt.setInt(1, id);
        prSt.executeUpdate();
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}
}

```

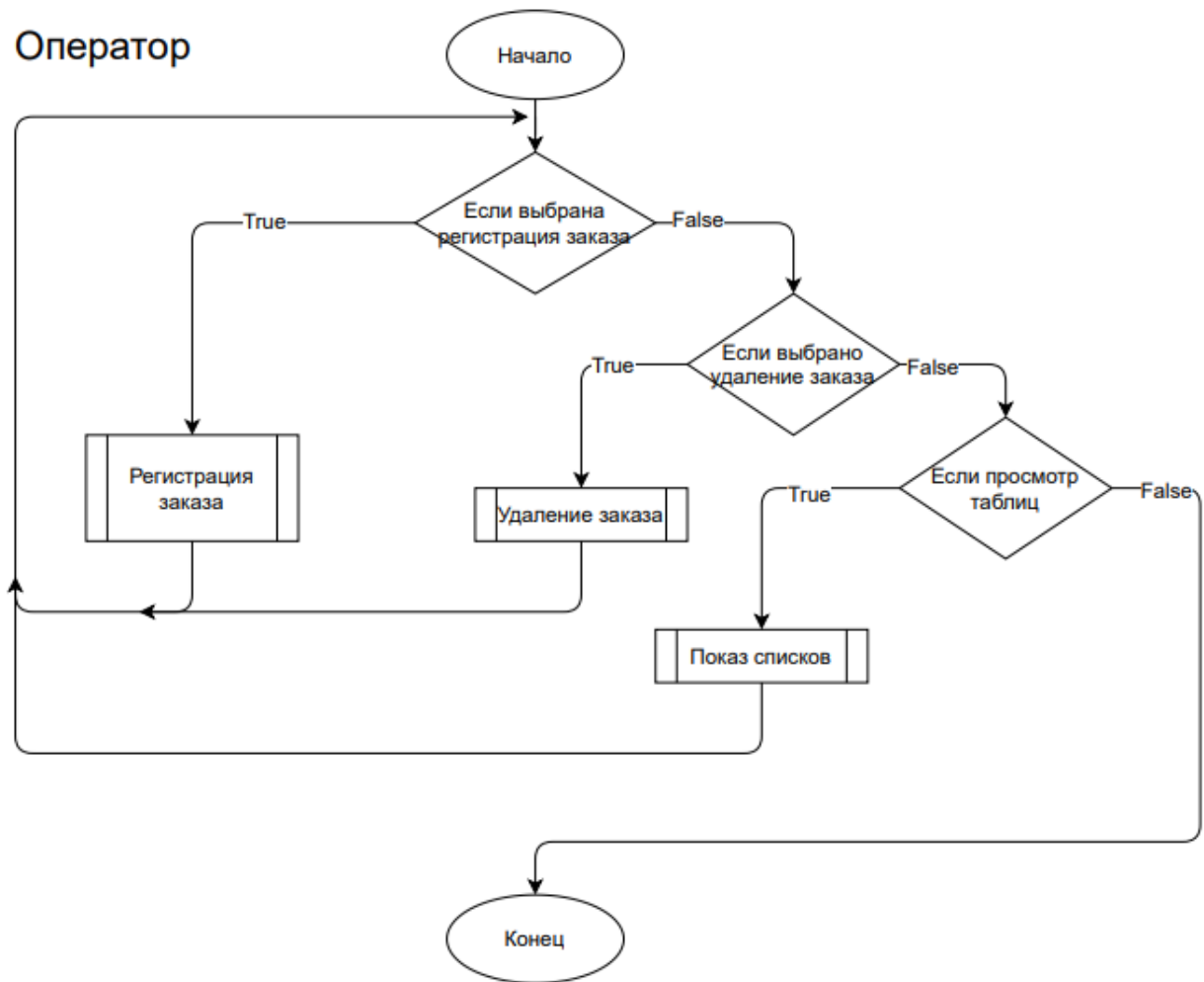
7. Укрупненные блок схемы.



Администратор

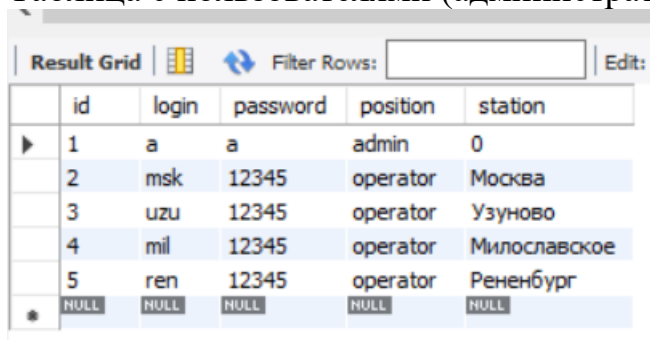


Оператор



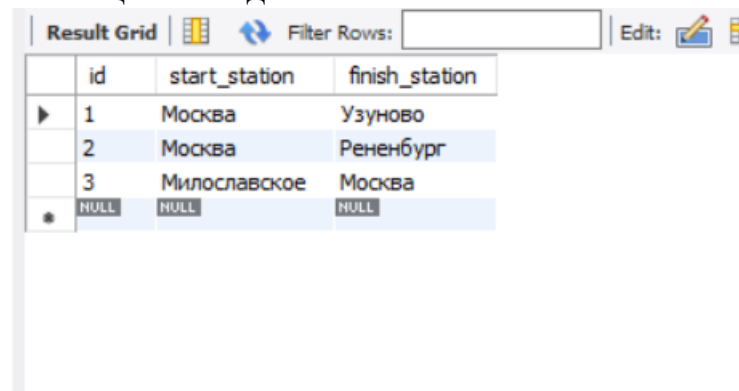
8. Скриншоты таблиц из СУБД.

Таблица с пользователями (администратор и операторы)



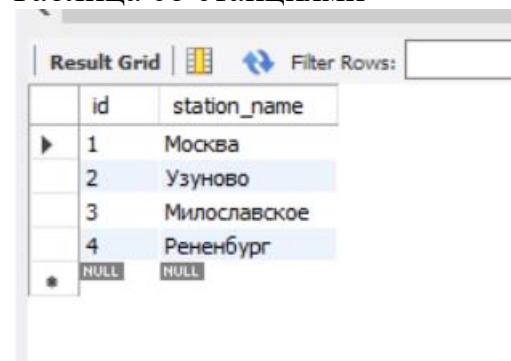
	id	login	password	position	station
▶	1	a	a	admin	0
	2	msk	12345	operator	Москва
	3	uzu	12345	operator	Узуново
	4	mil	12345	operator	Милославское
	5	ren	12345	operator	Рененбург
*	NULL	NULL	NULL	NULL	NULL

Таблица с поездами



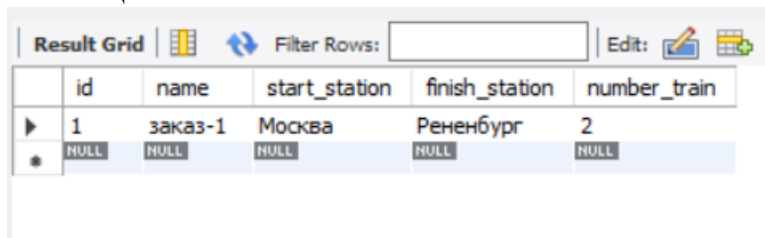
	id	start_station	finish_station
▶	1	Москва	Узуново
	2	Москва	Рененбург
	3	Милославское	Москва
*	NULL	NULL	NULL

Таблица со станциями



	id	station_name
▶	1	Москва
	2	Узуново
	3	Милославское
	4	Рененбург
*	NULL	NULL

Таблица с заказами



	id	name	start_station	finish_station	number_train
▶	1	заказ-1	Москва	Рененбург	2
*	NULL	NULL	NULL	NULL	NULL